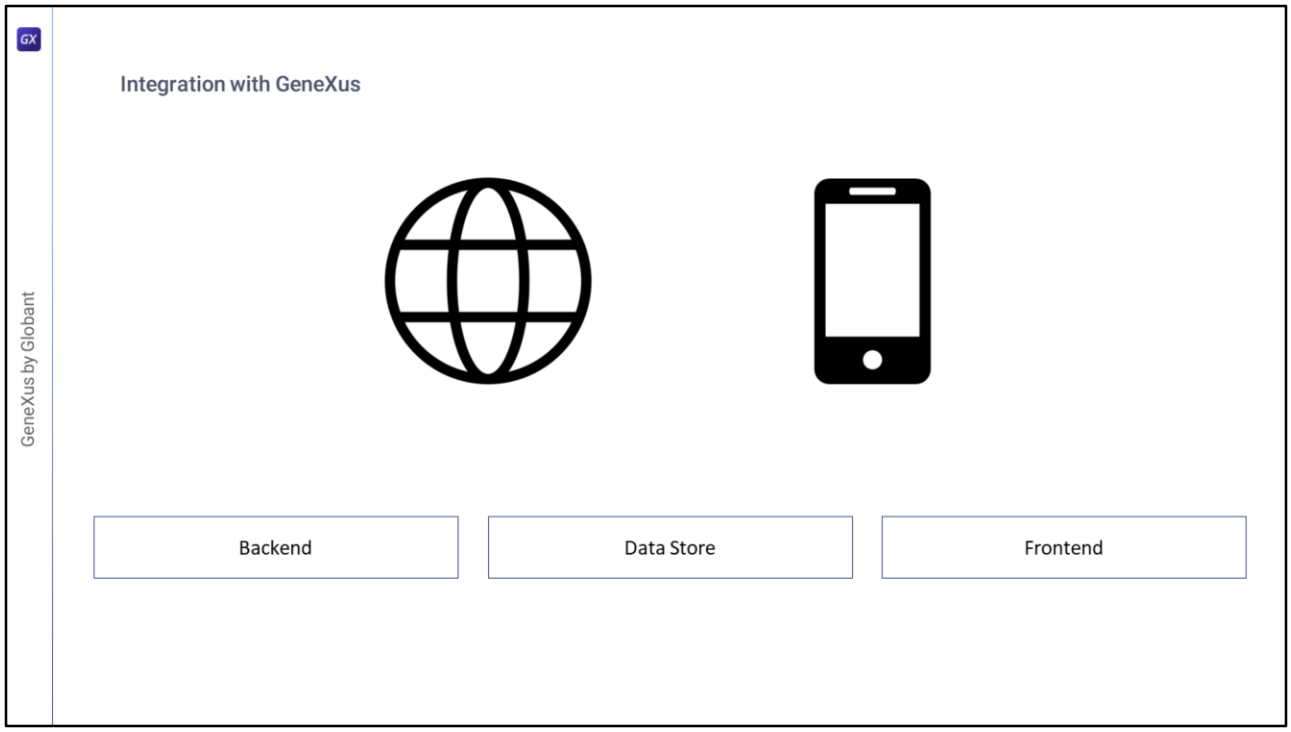

GeneXus by Globant
GeneXus™
by Globant
training.genexus.com

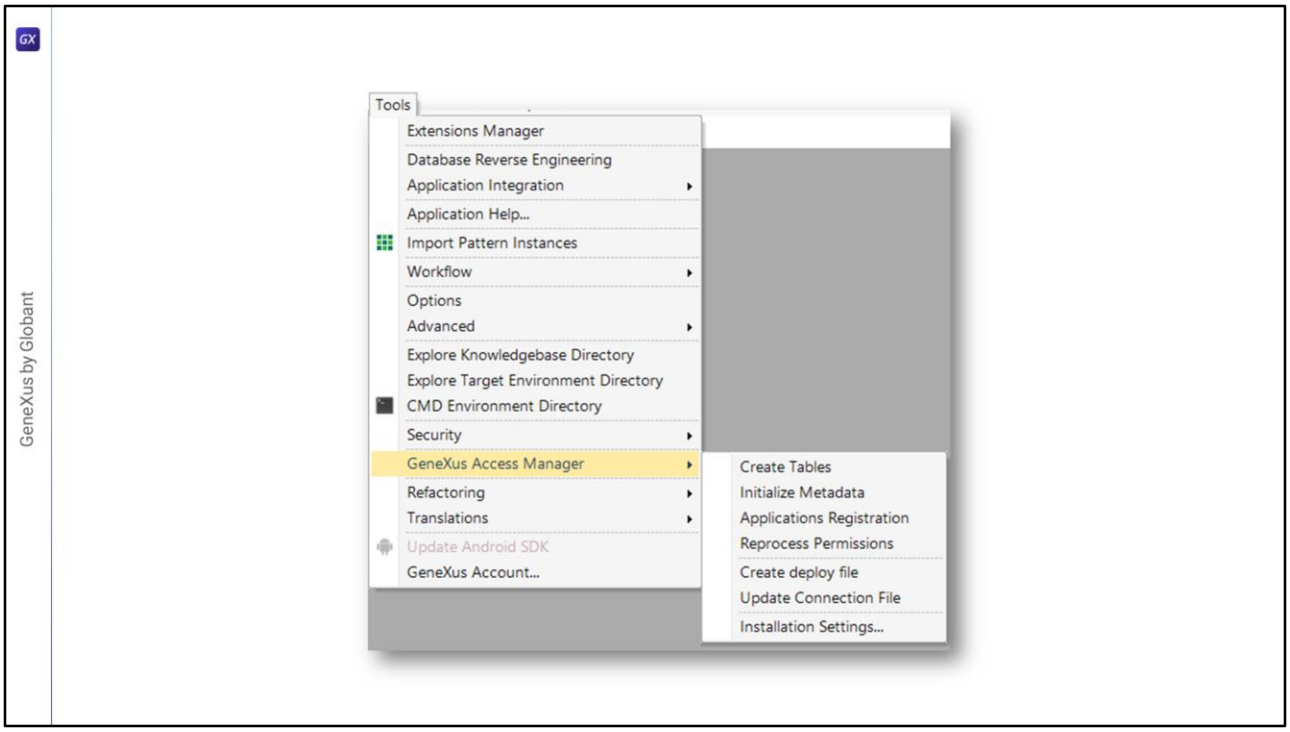
GeneXus by Globant

GAM Components

Nicolas Adrién



GAM supports platforms for running both web and mobile environments.
It is possible to define the language to be used in the back end, the database management systems, and the platform to be used in the front end. All this from the GeneXus UI.
Later on we will explore each one in depth.



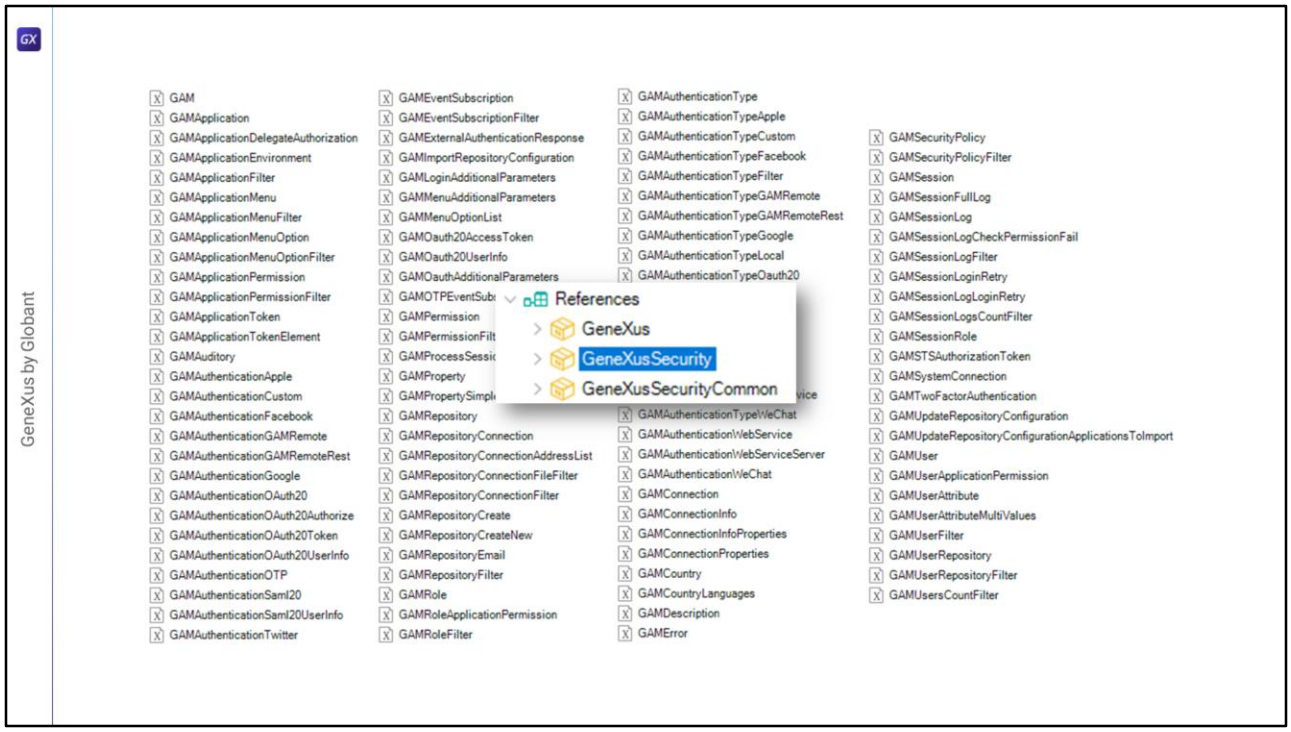
Bajo el menú Tools/GeneXus Access Manager, podemos acceder a las distintas funcionalidades que nos ofrece el GAM dentro de GeneXus.

Desde allí podemos:

- Crear las tablas que utiliza GAM
- Inicializar los metadatos
- Registrar las aplicaciones que utilizaran GAM
- Reprocesar los permisos de las mismas
- Crear el archivo de deploy
- Modificar el archivo de conexiones

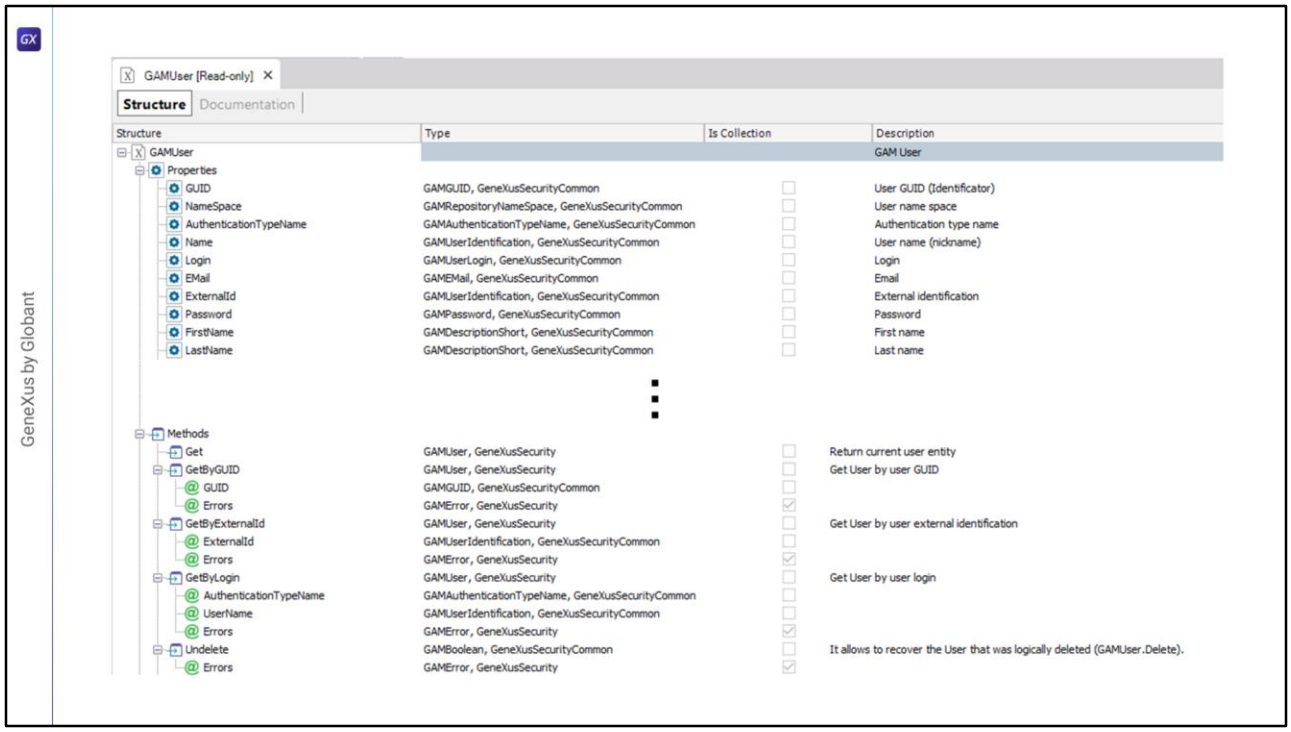
Y de ultimo,

- Configurar la instalación del mismo



In the References item of the context menu of the KB, inside GeneXusSecurity we can find all the External Objects provided by GAM, as shown on the screen.

Let's see the structure of one of them as an example.

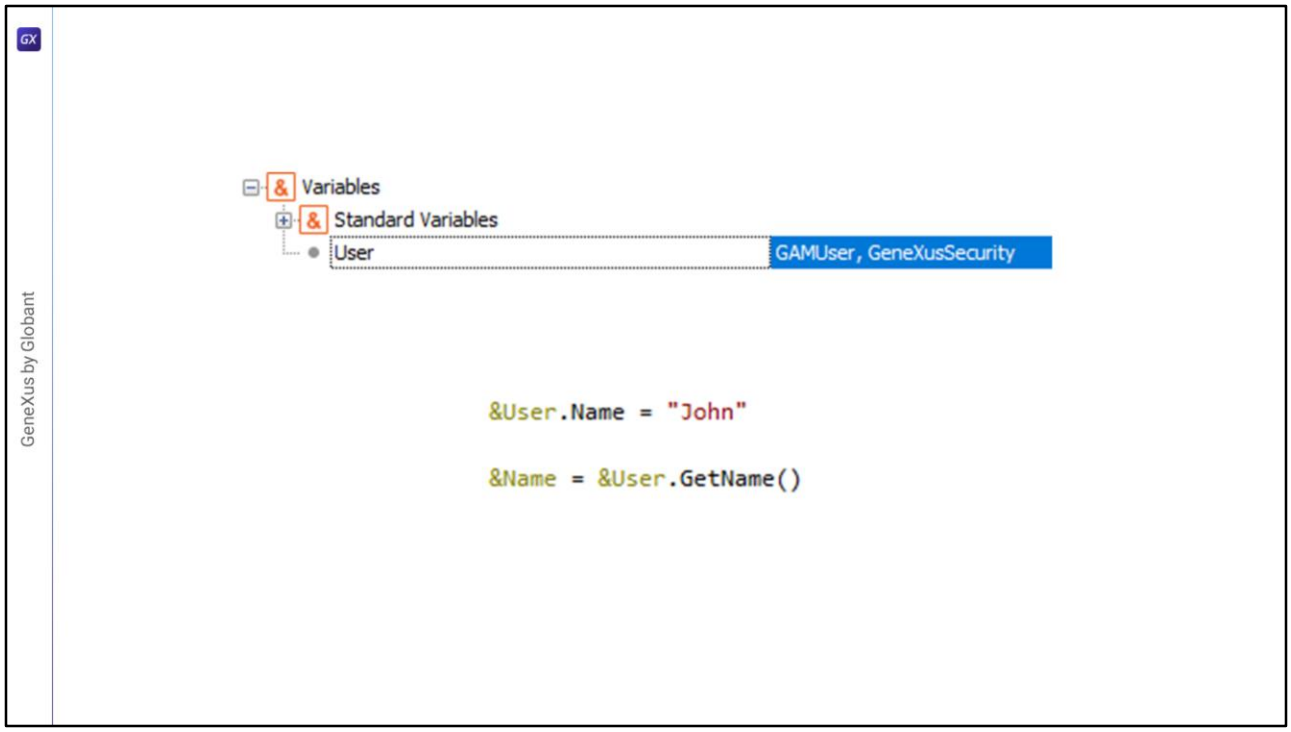


For example, a closer look at the GAMUser External Object reveals that its structure (like all the others) is composed of properties and methods.

Since this External Object is quite large, in this slide its content has been simplified and explained. From GeneXus, you can see all the properties and methods it contains.

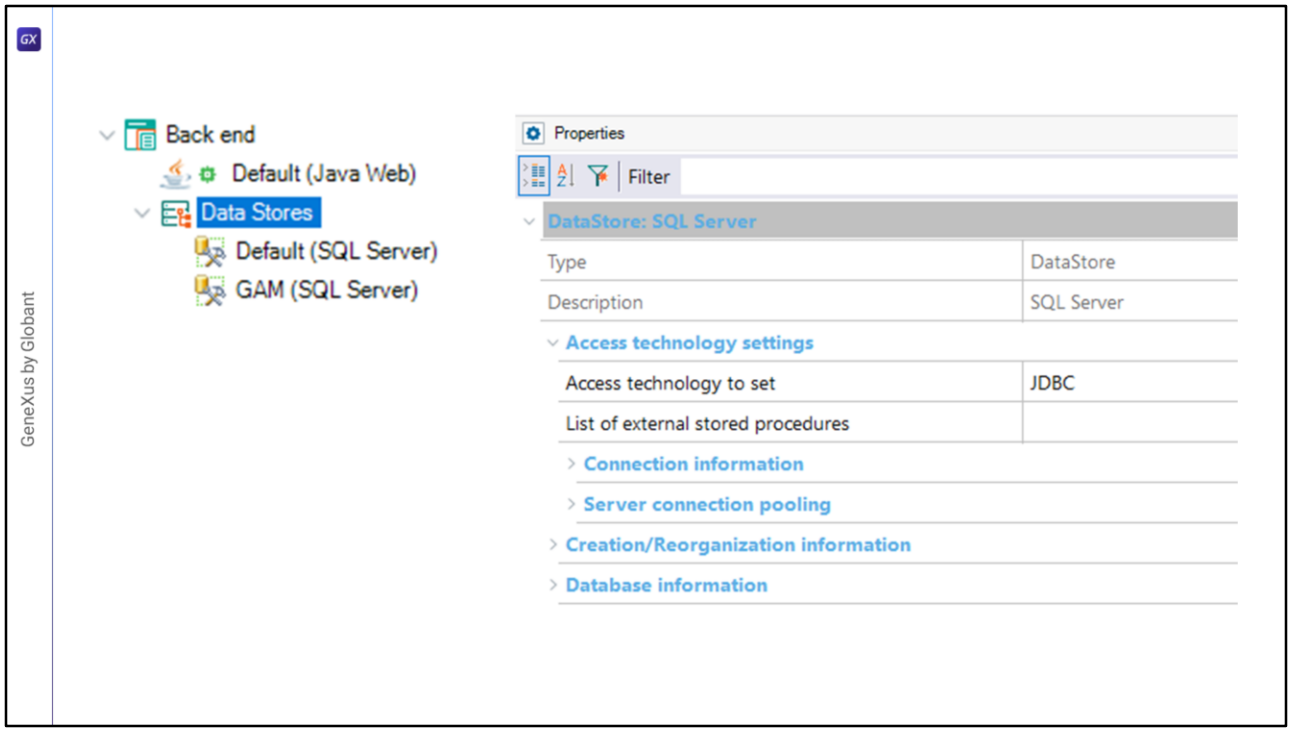
Note that for each property and method we have a type, whether it is a collection or not, and a description that will be helpful when using them.

Let's see an example of use.

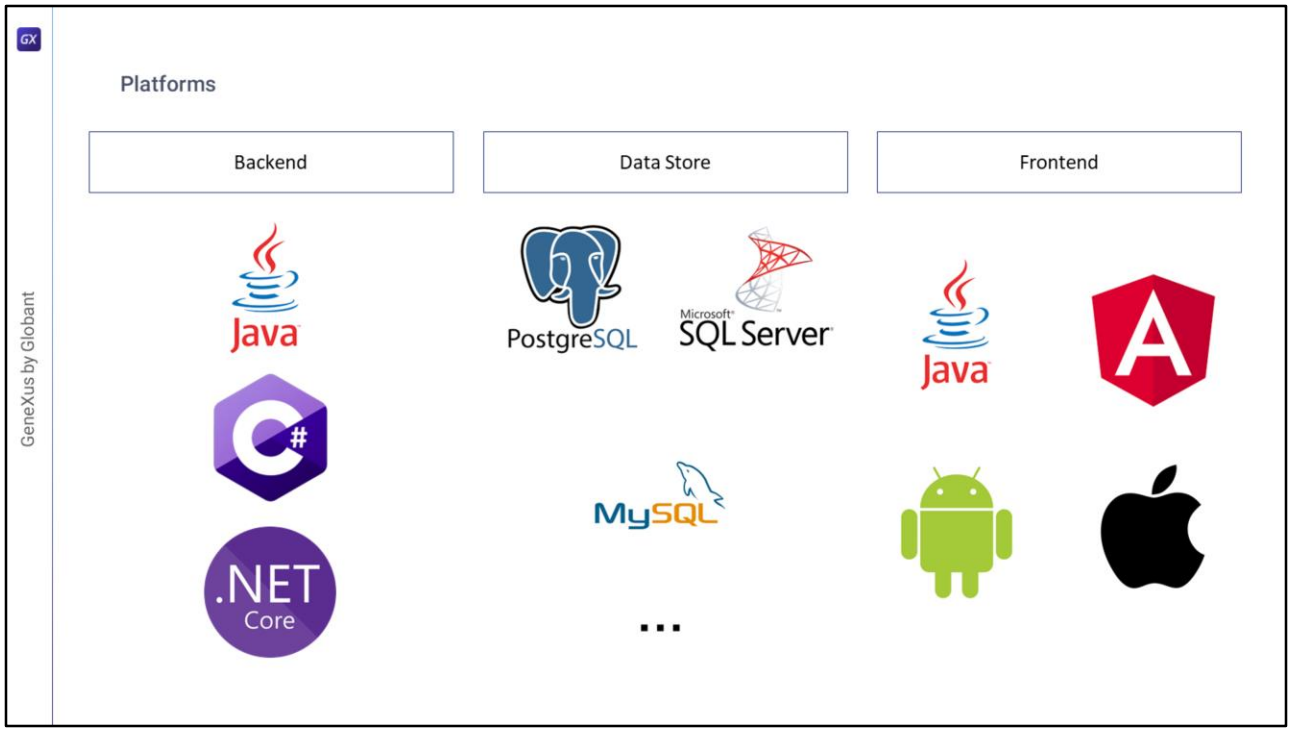


To use the External Object, we must create a variable of this type (GAMUser in this case). To use it in the code, values are simply assigned to its attributes or properties, or its methods are used to obtain the already loaded data.

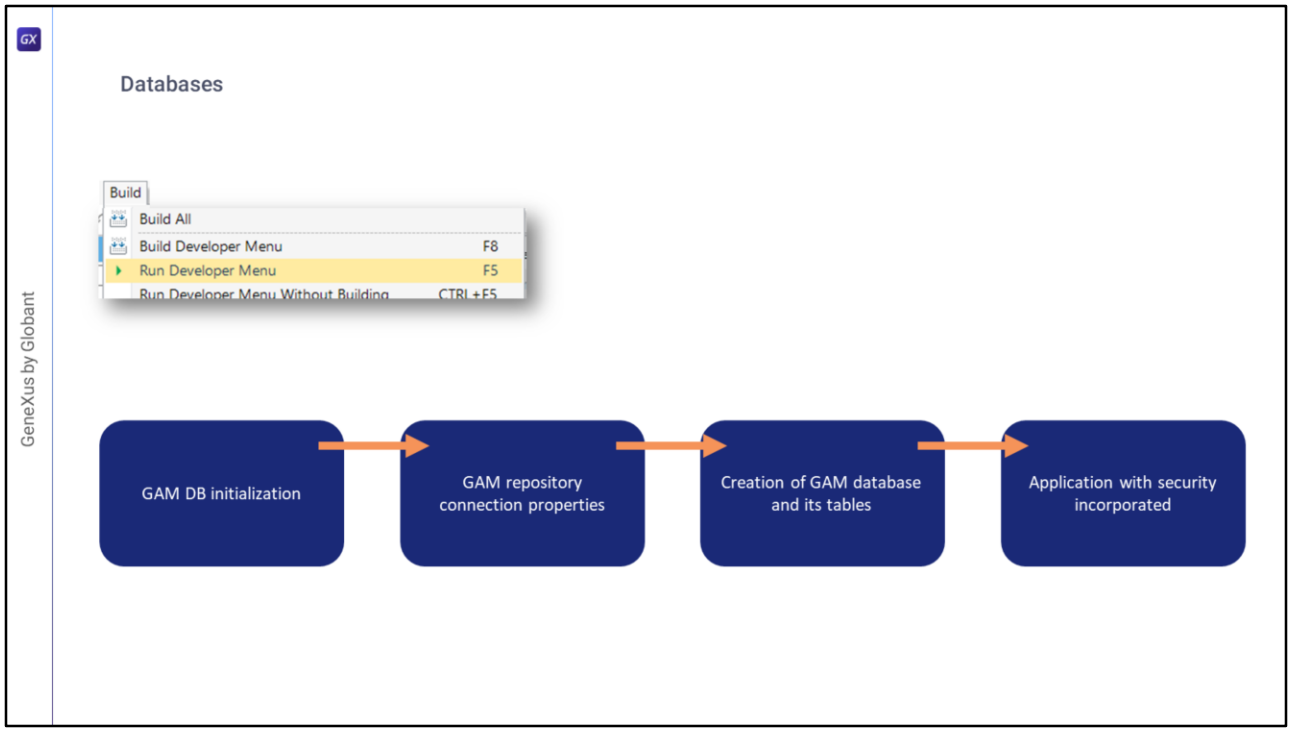
To use the rest of the External Objects, the same flow and methodology is followed.



When enabling GAM in the KB, a Data Store is generated for GAM to store its information together with that of the Application, where it is possible to configure the technology to be used, the connection and server data, etc.



Going into more details about the different platforms supported by GAM, we can say that:
The back end can be generated under Java, C#, and .NET Core languages.
For the Data Store, many options are offered including PostgreSQL, SQL Server, and MySQL.
Lastly, for the front end we can use web versions such as Java or Angular, and mobile versions such as Android and Apple.



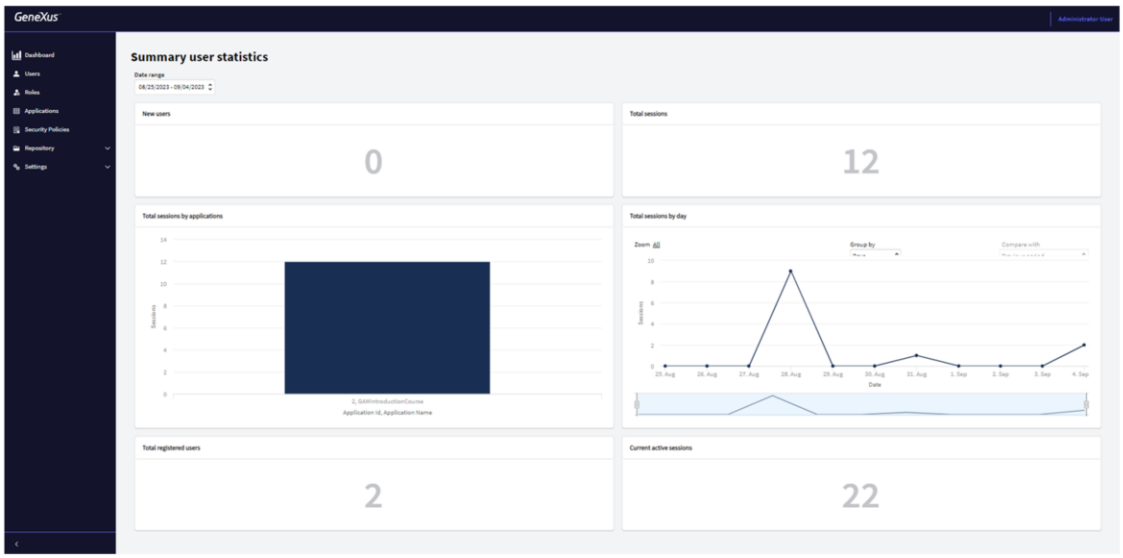
After running the Developer Menu, a connection to the database specified in the GAM Data Store is established, using its connection properties, and verifying the existence of some tables and the GAM version.

Since these tables do not exist, the GAM database tables are created. Then, the following happens:

First, some properties related to the connection to the GAM repository are assigned with their default values.
Next, the GAM database and all its tables are created. Before creating the tables, the user is asked if he/she wants to create the GAM database structure.

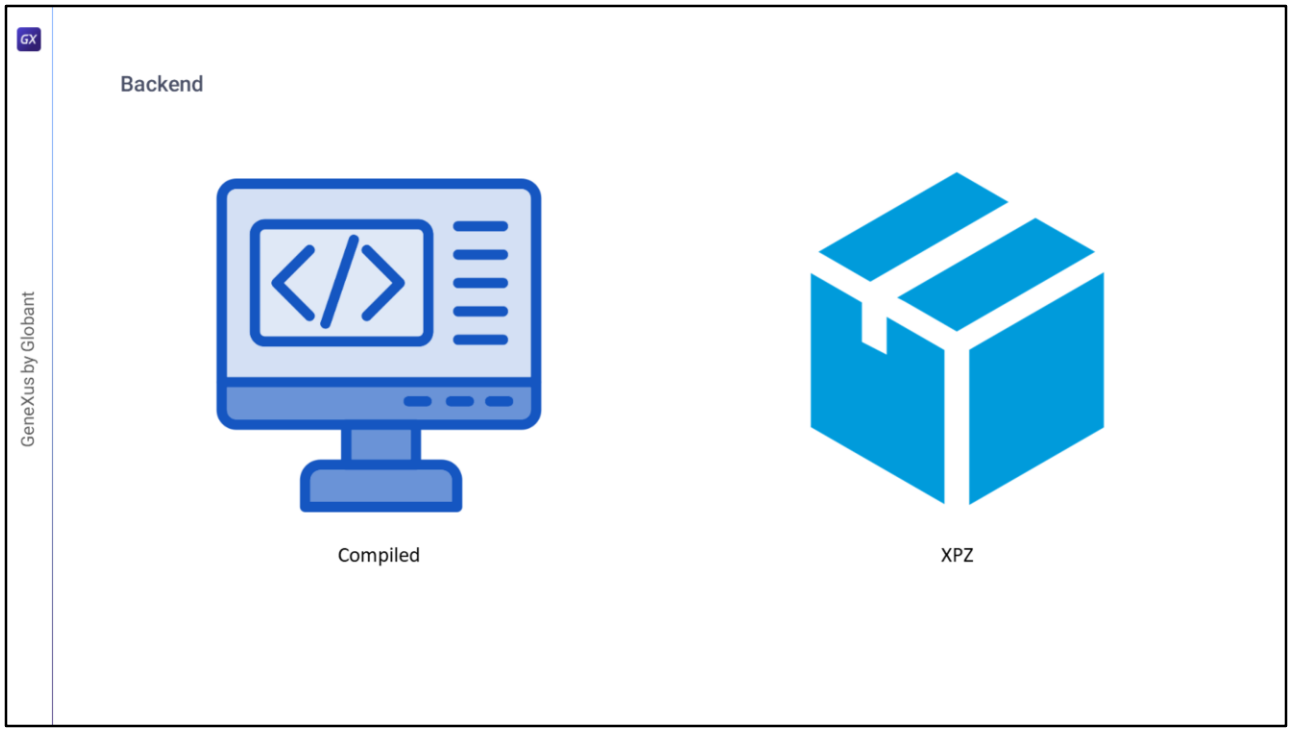
GeneXus by Globant

Backend



The GAM back end is a GAM Application automatically defined with the creation of the GAM metadata, during the initialization process.

The purpose of this application is to configure and manage all GAM related concepts.



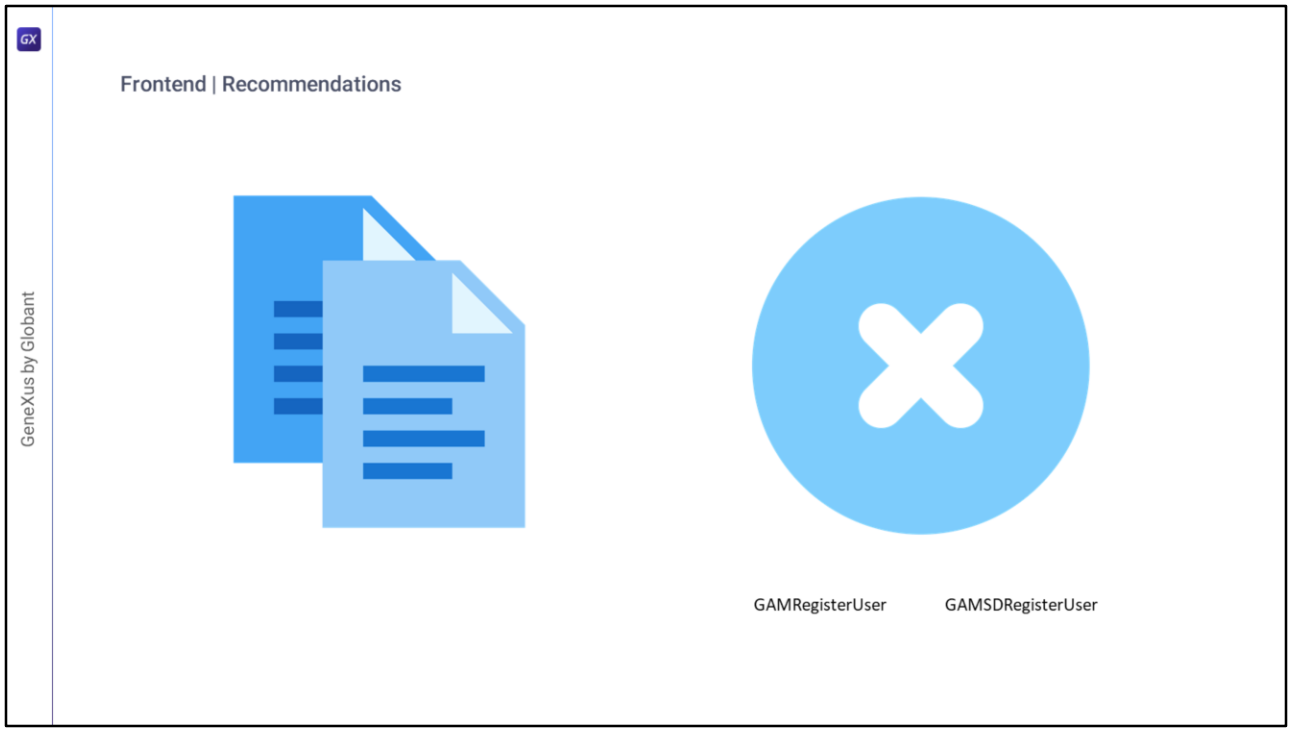
There are two versions of the back end.
The first one is the compiled one. By default, it is created when enabling GAM in the KB.
The second version is through an XPZ, which are the files used by the KB Manager to exchange objects between knowledge bases. The name is derived from the extension of these files. In this case, GAM can be incorporated into the KB by importing this xpz included in the GeneXus installation.

These objects are useful because they are examples of GAM API usage.
They can be modified as desired by the GeneXus developer if some requirements are not met (the GAM API is available for this purpose).

When GAM is enabled, all the example objects from the front end are imported into the KB.

The GAM examples are web objects and SD panels that use the GAM API, and their purpose is to help the GeneXus user learn how to use this API.
Another important purpose of these objects is to help the user get started with GAM, since they include objects for login, registration, password change, and redirection in case of Authorization error, among others.

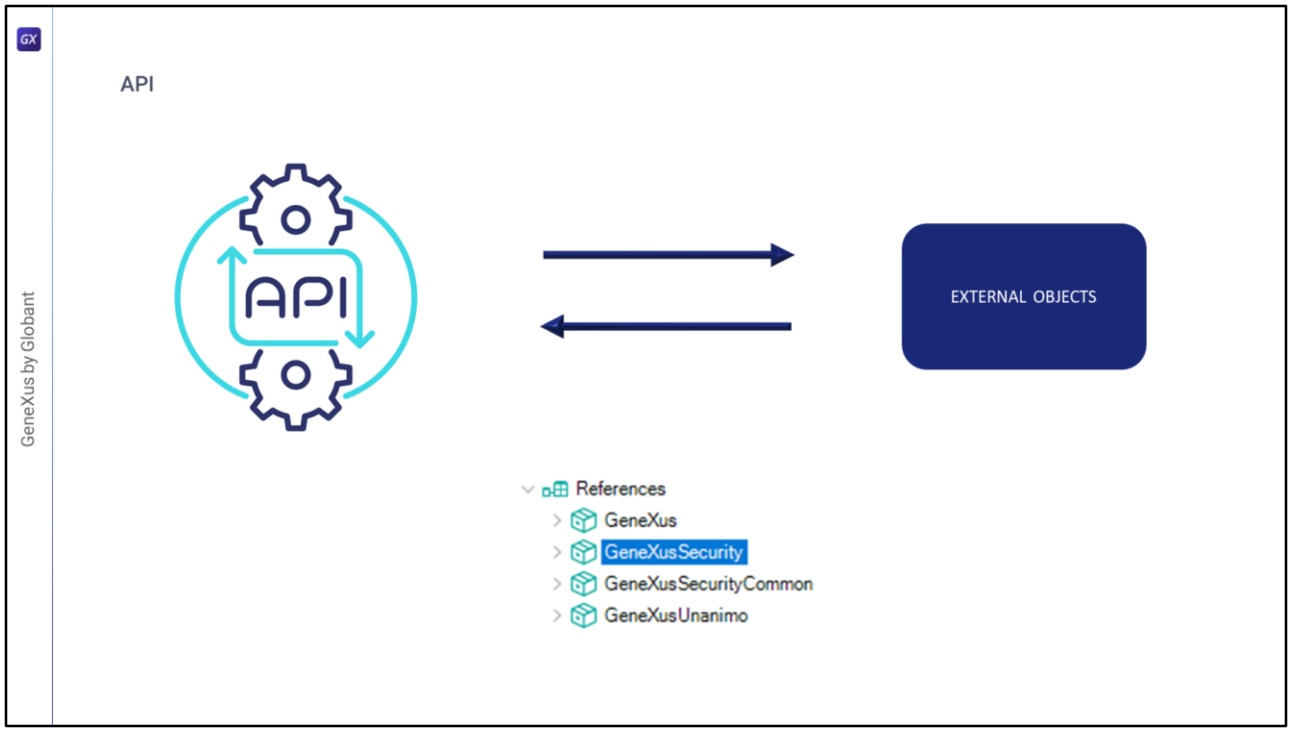
All these objects are consolidated in the KB during the GAM activation process and located in the GAM Examples folder; every time a new GAM build or upgrade is installed, the user can decide whether to update these examples with the look and feel of the application, for example.



It is good practice to create copies of the examples to be used in the application. In this way, when GeneXus (and therefore GAM) is updated, the changes introduced will not be overwritten.

Another recommendation is to delete those panels that will not be used. One example of this is the self-registration that we will see later.

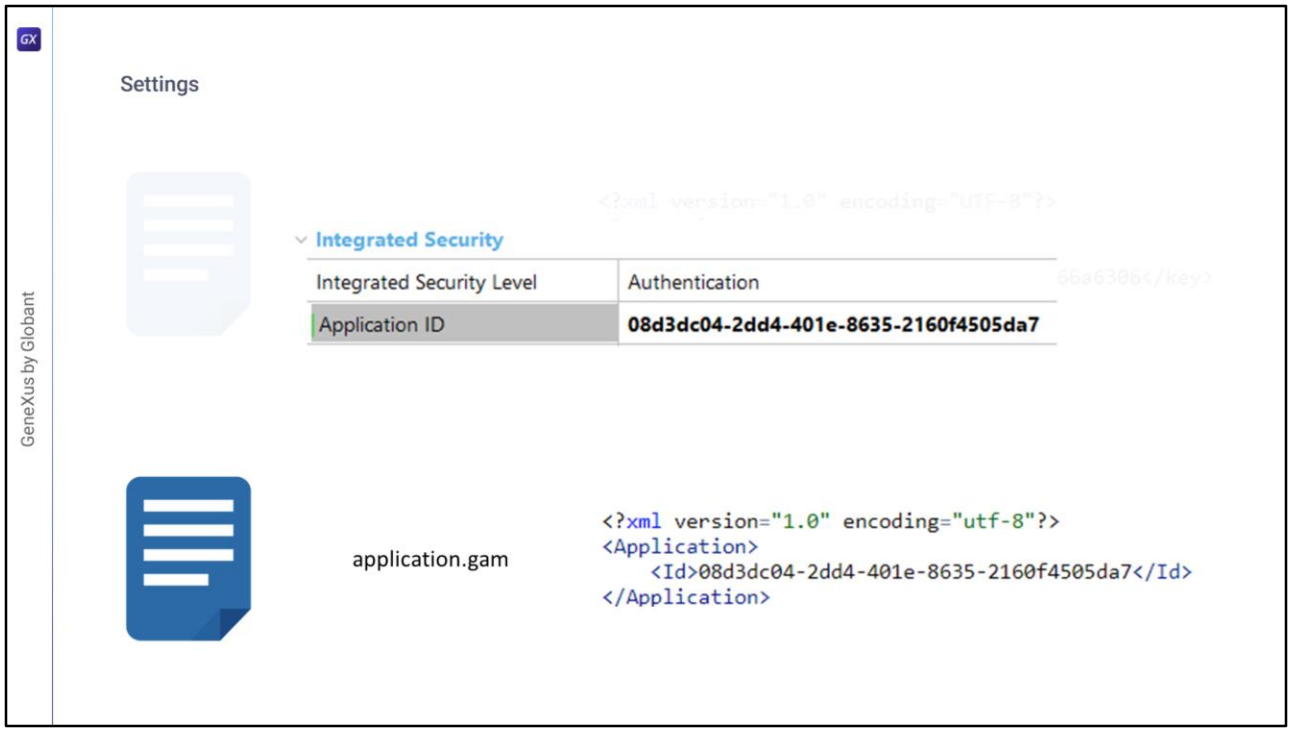
If users will be registered in the application by an administrator, it is recommended to delete the GAMRegisterUser and GAMSDRegisterUser panels.



GAM provides an API that allows users to handle data types and methods to add security to GeneXus applications (both web and mobile applications).

When integrated security is enabled in the KB, external objects are incorporated to allow the user to interact with the GAM API. External objects are the way to access the GAM API. They are all distributed in a module called GeneXusSecurity, as we said before.

APIs are those used by the example panels. From them, it is possible to perform all the actions available in the GAM Backend.



GAM generates different files with configurations for connection to repositories.
These files are connection.gam and application.gam.

connection.gam is a GAM configuration file that contains the key associated with each Repository to which you want to connect. The connection data associated with the key must exist in the GAM database (it must have been previously created using the GAM API).

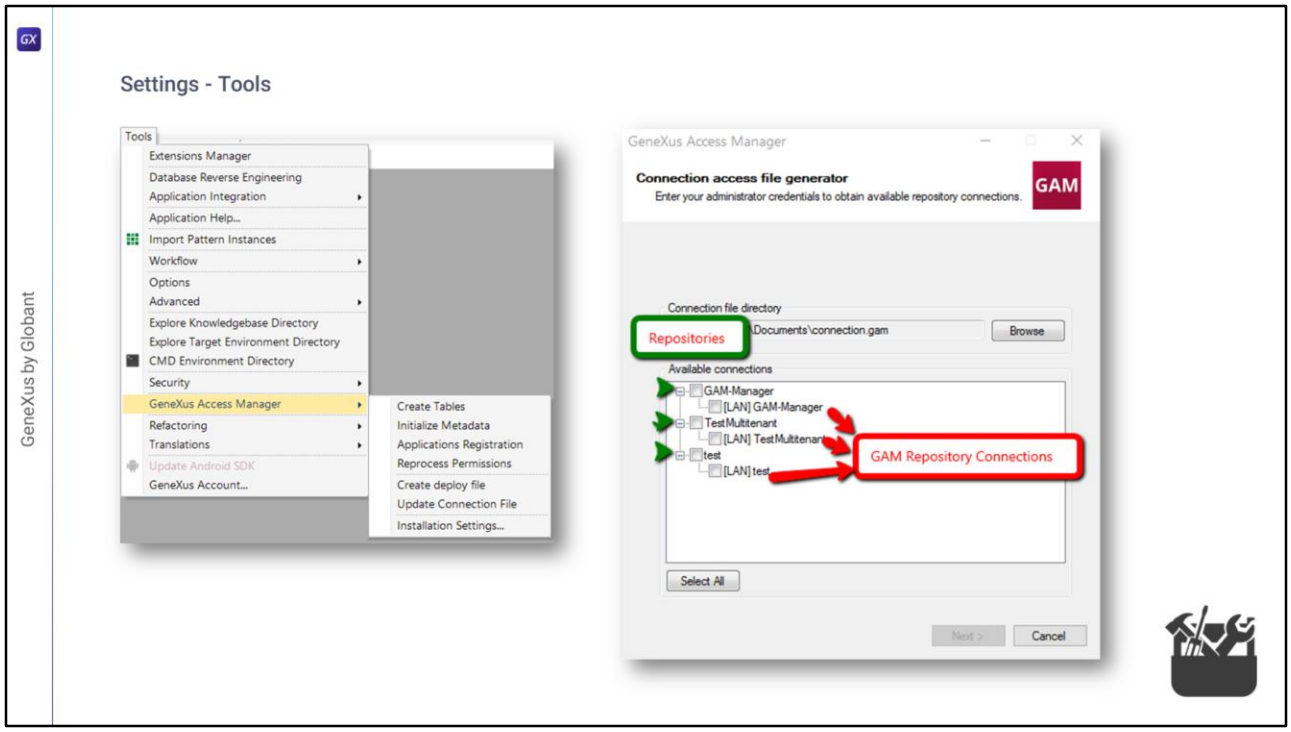
In the case of Java, this file is copied to the root of the web application on the servlet server (from Evolution 2 upgrade 3 onwards).
In the case of NET, it goes in the virtual directory (that is, the web directory).

application.gam is a GAM configuration file that contains the GAM WEB Application ID of the KB specified in the Application ID property, and is used for GAM to identify which web application is running.

At prototyping time, it is automatically transferred to the WEB-INF folder for Java applications and to the virtual directory for NET applications.

When taking the application to production, this file must be included in the deployment.

The ID contained in it must exist in an application defined in the GAM Database, and can be configured from the GeneXus IDE through the Environment properties in the Integrated Security section, as shown on the screen.



There are GAM tools to modify the information of the configuration files for different environments, such as Testing, Pre-Production or Production, in order to make it easier to edit them.

For the **connection.gam file**, there is the **GamDeployTool** that can be executed from the GeneXus IDE, by selecting the *GAM - Update Connection File* option.

This tool is intended for use by the GAM Manager Repository administrators.

When it is used, entries are added to certain GAM database tables depending on the user's selection, which can be made among the existing GAM Repository connections of each Repository.

At the end of the process, the file is generated at the specified address and then copied to the web application.

Other types of configuration files are **client.cfg** and **web.config**.

They depend on the generator being used:

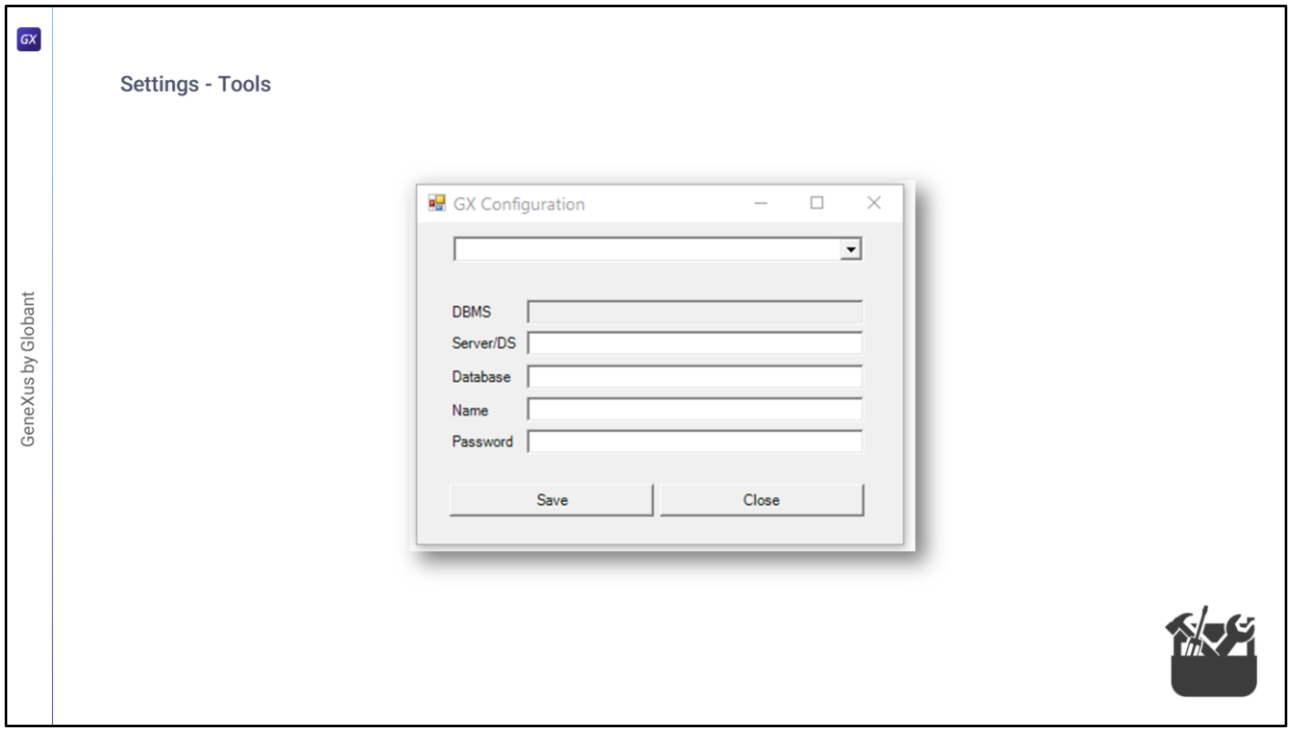
In Java applications, client.cfg is used (where it applies to web and 2-tier applications).

In .Net applications, web.config is used.

For the **client.cfg** and **web.config**, at the time of enabling GAM in a knowledge base, the lines shown on the screen are set in them.

They correspond to setting the integrated security to 1, and defining the Login and Unauthorized example objects.

In addition, a new DataSource corresponding to GAM is added, leaving the application DEFAULT in second place.



As for the tools for these last two files, we have:

For the **web.config** file, there is a tool made by GeneXus called GxConfig, which facilitates the creation of the file through the graphical interface we see on screen.

For the **client.cfg** file, there are different ways to facilitate the configuration of the file that vary according to the GeneXus version being used.

In several scenarios, the common practice is to read configuration data from environment variables, rather than from configuration files.

Since upgrade #5 of GeneXus 17, the GAM connection key can be specified through an environment variable called GX_GAMCONNECTIONKEY.
This allows, for example, that when deploying to Docker you can configure the connection key to be used for that instance of the container.

When trying to access any GAM API from an application, the first thing to do is to check if this environment variable is configured to obtain the GAM connection data.
If it is not, the connection.gam file is searched to obtain this Key.

The GeneXus Wiki contains a document describing the scenarios in which environment variables are used, including their advantages and how to do so.


GeneXus by Globant
GeneXus™
by Globant
training.genexus.com